(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－222374

(43)公開日　平成10年(1998)８月21日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/445		Ｇ０６Ｆ　9/06	４２０Ｍ

審査請求　未請求　請求項の数32　ＯＬ　外国語出願　（全 48 頁）

(21)出願番号　特願平9－294544

(22)出願日　平成９年(1997)10月27日

(31)優先権主張番号　６０／０２９２７７

(32)優先日　1996年10月28日

(33)優先権主張国　米国（ＵＳ）

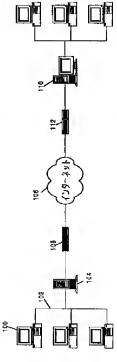
(71)出願人　597154922
アルテラ　コーポレイション
Ａｌｔｅｒａ　Ｃｏｒｐｏｒａｔｉｏｎ
アメリカ合衆国　95134　カリフォルニア州　サン　ホセ　イノベーション　ドライヴ　101

(72)発明者　ティモシー　ジェイ．サウスゲート
アメリカ合衆国　94062　カリフォルニア州　レッドウッド　シティ　ウイルミントン　ウェイ　908

(74)代理人　弁理士　恩田　博宣

(54)【発明の名称】　遠隔ソフトウェア技術支援を提供するための方法

(57)【要約】
【課題】遠隔地からソフトウェアベンダが顧客を把握し、新規バージョンを用いて顧客のソフトウェアをアップデートし、また、ソフトウェアのバグを修正することのできる技術を提供する。
【解決手段】顧客プラットホーム上のソフトウェアを遠隔的にアップグレードする方法は、顧客プラットホーム上のソフトウェアの実行に応じて顧客プラットホームとベンダプラットホームとの間にインターネット接続を確立する。顧客ソフトウェアに関連する第１バージョンのデータがベンダプラットホームからの第２バージョンのデータと異なる場合、第１のソフトウェアの第１の部分をベンダプラットホームからの第１アップデートコードで上書きする。



【特許請求の範囲】
【請求項１】第１のソフトウェアをアップグレードするための方法であって、
第１のプラットホーム上の前記第１のソフトウェアの実行に応じて前記第１のプラットホームと遠隔プラットホームとの間に接続を確立する工程と、
前記第１のソフトウェアに関連する第１バージョンのデータが前記遠隔プラットホームからの第２バージョンのデータと異なる場合、第１のソフトウェアの第１の部分を前記遠隔プラットホームからの第１アップデートコードで上書きする工程とを備える、方法。
【請求項２】請求項１に記載の方法において、前記第１のソフトウェアは複数のソフトウェアモジュールを備え、前記上書き工程は前記第１バージョン及び前記第２バージョンのデータが異なる前記複数のソフトウェアモジュールの各々について実行される、方法。
【請求項３】請求項１に記載の方法はさらに、前記遠隔プラットホームからの既知のエラーデータが前記第１のソフトウェアと一致する場合には、前記第１のソフトウェアの第２の部分を第２アップデートコードで上書きする工程を備える、方法。
【請求項４】請求項３に記載の方法はさらに、前記既知エラーデータを前記遠隔プラットホームから前記第１プラットホームに対してダウンロードする工程を備える、方法。
【請求項５】請求項１に記載の方法において、前記接続確立工程はインターネット接続の確立を含む、方法。
【請求項６】請求項５に記載の方法はさらに、前記インターネットに対するアクセスが利用可能であるか否かを決定する工程を備える、方法。
【請求項７】請求項１に記載の方法はさらに、前記接続を確立するために前記第１のソフトウェアのユーザから承認を取得する工程を備える、方法。
【請求項８】請求項７に記載の方法はさらに、承認が得られない場合、前記第１のソフトウェアの前記ユーザに対してソフトウェア支援情報を提供する工程を備える、方法。
【請求項９】請求項１に記載の方法はさらに、前記第１のソフトウェアの前記第１の部分を上書きする前に前記第１のソフトウェアの実行をシャットダウンする工程を備える、方法。
【請求項１０】請求項１に記載の方法はさらに、前記第１のソフトウェアの前記第１の部分を上書きした後、ソフトウェア診断ルーチンを前記第１のソフトウェア上で実行する工程を備える、方法。
【請求項１１】請求項１に記載の方法はさらに、前記第１バージョンのデータを前記遠隔プラットホームから前記第１のプラットホームに対してダウンロードする工程を備える、方法。
【請求項１２】第１のソフトウェアをアップグレードするためのプログラム命令を記録した１つ以上のコンピュータ読取可能媒体であって、
第１のプラットホーム上の前記第１のソフトウェアの実行に応じて前記第１のプラットホームと遠隔プラットホームとの間に接続を確立するためのコンピュータ読取可能コードと、
前記第１のソフトウェアに関連する第１バージョンのデータが前記遠隔プラットホームからの第２バージョンのデータと異なる場合、第１のソフトウェアの第１の部分を前記遠隔プラットホームからの第１アップデートコードで上書きするためのコンピュータ読取コードとを備える、コンピュータ読取可能媒体。
【請求項１３】搬送波に組み込まれるコンピュータデータ信号であって、１個以上の演算処理装置により実行される際、
第１のプラットホーム上の前記第１のソフトウェアの実行に応じて前記第１のプラットホームと遠隔プラットホームとの間に接続を確立し、
前記第１のソフトウェアに関連する第１バージョンのデータが前記遠隔プラットホームからの第２バージョンのデータと異なる場合、第１のソフトウェアの第１の部分を前記遠隔プラットホームからの第１アップデートコードで上書きすることにより、前記１個以上の演算処理装置に前記第１のソフトウェアをアップグレードさせる連続命令を表すコンピュータデータ信号。
【請求項１４】第１のソフトウェアに対応するプログラム命令を記録したコンピュータ読取媒体であって、前記第１のソフトウェアは、
第１のプラットホーム上の前記第１のソフトウェアの実行に応じて前記第１のプラットホームと遠隔プラットホームとの間に接続を確立する工程と、
前記第１のソフトウェアに関連する第１バージョンのデータが前記遠隔プラットホームからの第２バージョンのデータと異なる場合、第１のソフトウェアの第１の部分を前記遠隔プラットホームからの第１アップデートコードで上書きする工程とを備える方法に従いアップグレードされる、コンピュータ読取可能媒体。
【請求項１５】第１のソフトウェアの問題を解決するための方法であって、
第１のプラットホーム上の前記第１のソフトウェアの実行エラーに応じて前記第１のプラットホームと遠隔プラットホームとの間に接続を確立する工程と、
前記エラーが前記遠隔プラットホームからの既知のエラーデータと一致する場合、第１のソフトウェアの部分をアップデートコードで上書きする工程とを備える、方法。
【請求項１６】請求項１５に記載の方法はさらに、前記既知エラーデータを前記遠隔プラットホームから前記第１プラットホームに対してダウンロードする工程を備える、方法。

【請求項１７】請求項１５に記載の方法において、前記接続確立工程はインターネット接続の確立を含む、方法。
【請求項１８】請求項１７に記載の方法はさらに、前記インターネットに対するアクセスが利用可能であるか否かを決定する工程を備える、方法。
【請求項１９】請求項１５に記載の方法はさらに、前記接続を確立するために前記第１のソフトウェアのユーザから承認を取得する工程を備える、方法。
【請求項２０】請求項１９に記載の方法はさらに、承認が得られない場合、前記第１のソフトウェアの前記ユーザに対してソフトウェア支援情報を提供する工程を備える、方法。
【請求項２１】請求項１５に記載の方法はさらに、前記第１のソフトウェアの前記第１の部分を上書きする前に前記第１のソフトウェアの実行をシャットダウンする工程を備える、方法。
【請求項２２】請求項１５に記載の方法はさらに、前記第１のソフトウェアの前記第１の部分を上書きした後、ソフトウェア診断ルーチンを前記第１のソフトウェア上で実行する工程を備える、方法。
【請求項２３】請求項１５に記載の方法はさらに、前記エラーが既知のエラーデータと一致しない場合、前記接続を介して前記第１のソフトウェアに関連するデータファイルを前記第１のプラットホームから前記遠隔プラットホームに対してアップロードする工程と、
前記データファイルを用いて前記遠隔プラットホーム上で前記エラーを再現する工程とを備える、方法。
【請求項２４】請求項２３に記載の方法はさらに、前記データファイルをアップロードするために前記第１のソフトウェアのユーザから承認を取得する工程を備える、方法。
【請求項２５】請求項２４に記載の方法はさらに、承認が得られない場合、前記第１のソフトウェアの前記ユーザに対してソフトウェア支援情報を提供する工程を備える、方法。
【請求項２６】請求項２３に記載の方法はさらに、前記データファイルをアップロードする前に、前記データファイルに関する前記第１のプラットホームからの情報をコンパイルする工程を備える、方法。
【請求項２７】請求項１５に記載の方法はさらに、前記エラーが既知のエラーデータと一致しない場合、前記接続を介して前記遠隔プラットホームから前記第１のプラットホーム上において前記第１のソフトウェアを実行する工程を備え、これにより前記エラーが前記第１のプラットホーム上で再現される、方法。
【請求項２８】請求項２７に記載の方法はさらに、前記遠隔プラットホームから前記第１のソフトウェアを実行するために前記第１のソフトウェアのユーザから承認を取得する工程を備える、方法。
【請求項２９】請求項２８に記載の方法はさらに、承認が得られない場合、前記第１のソフトウェアの前記ユーザに対してソフトウェア支援情報を提供する工程を備える、方法。
【請求項３０】第１のソフトウェアの問題を解決するためのプログラム命令を記録した１つ以上のコンピュータ読取可能媒体であって、
第１のプラットホーム上の前記第１のソフトウェアの実行エラーに応じて前記第１のプラットホームと遠隔プラットホームとの間に接続を確立するためのコンピュータ読取可能コードと、
前記エラーが前記遠隔プラットホームからの既知のエラーデータと一致する場合、第１のソフトウェアの部分をアップデートコードで上書きするためのコンピュータ読取可能コードとを備える、コンピュータ読取可能媒体。
【請求項３１】搬送波に組み込まれるコンピュータデータ信号であって、１個以上の演算処理装置により実行される際、
第１のプラットホーム上の前記第１のソフトウェアの実行エラーに応じて前記第１のプラットホームと遠隔プラットホームとの間に接続を確立し、
前記エラーが前記遠隔プラットホームからの既知のエラーデータと一致する場合、第１のソフトウェアの部分をアップデートコードで上書きすることにより前記１個以上の演算処理装置に前記第１のソフトウェアの問題を解決させる連続命令を表すコンピュータデータ信号。
【請求項３２】第１のソフトウェアに対するプログラム命令を記録したコンピュータ読取媒体であって、前記第１のソフトウェアは、
第１のプラットホーム上の前記第１のソフトウェアの実行エラーに応じて前記第１のプラットホームと遠隔プラットホームとの間に接続を確立する工程と、
前記エラーが前記遠隔プラットホームからの既知のエラーデータと一致する場合、第１のソフトウェアの部分をアップデートコードで上書きする工程とを備える方法に従い問題解決される、コンピュータ読取可能媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、例えば、インターネットを介する遠隔地からのソフトウェアの技術支援を提供するための方法に関する。より詳細には、本発明は、それによりソフトウェアのアップデート、及びソフトウェアのバグの修正が遠隔地から顧客のソフトウェア中に組み込まれ得る方法を提供する。
【０００２】
【従来の技術】一旦、ソフトウェア製造会社がソフトウェア製品をその顧客に対して販売すると、これらの顧客を把握し続け、そして、ソフトウェアを定期的にアップデートすることが非常に重要となる。バグのないソフトウェアは存在し得ず、事実、ソフトウェア産業は、ソフ

トウェアのエラーを修正するために、定期的にソフトウェアプログラムの新規バージョンを再引渡しすることにかなり慣れている。これらの後の引渡は、ソフトウェアに対して新規機能を付加し、小さな問題を修正し、あるいは、しばしばソフトウェアの重大なエラーを修正する。ソフトウェアプログラムの性能は顧客にとって重要なので、ソフトウェア会社の最大の関心事項は、ソフトウェアのこれら新バージョンを顧客に対して引渡し、そして、新規ソフトウェアバージョンが、正しく受け取られると共にインストールされることにある。

【０００３】

【発明が解決しようとする課題】しかしながら、ソフトウェア会社が最大限努力しても、自社の全顧客のための新規バージョンを用いた自社ソフトウェア製品の定期的なアップデートを妨げる多くの障害が存在する。問題の１つは、ソフトウェアプログラムを購入する顧客の内、プログラムの登録、及び登録のソフトウェア会社への返送という正規手続きをやり終える者はほとんどいないことにある。登録用紙により、ソフトウェア会社は、自社のソフトウェアの購入者、購入者の住所、担当者、購入バージョン等を把握することができる。しかしながら、そのようなソフトウェア登録用紙はソフトウェアベンダによってほとんど受け取られることはない。このことは、定期的なアップデートがだれに対して送られるべきかというソフトウェア会社の決定を非常に困難にする。ソフトウェアを定期的（例えば、年に４回）にアップデートしなければならない、又は、重大なソフトウェアのバグを修正しなければならない可能性のあるそれらソフトウェア会社にとって、このソフトウェアの登録の欠如は真の問題をもたらす。登録用紙が受け取られていたとしても、ソフトウェアの新規バージョンを誰に対して送るべきかを正確に解決することはしばしば困難である。組織内での個人の部署の異動、組織の合併、エンジニアの離職等は全て、組織内の誰に対してソフトウェアの新規バージョンを送るべきかの決定を非常に困難なものにする。ソフトウェアを実行する全てのコンピュータが新規バージョンを用いてアップデートされ、インストールが適切に実行されることを確実にするため、組織内の責任者に対してソフトウェアの新規バージョンが送られることも重要である。しかしながら、そのような適任者であってもインストールに際して過ちを犯す。

【０００４】ソフトウェアの新規バージョンを顧客に対して引渡す際における他の問題は、その配布にある。ソフトウェアの新規バージョンを引き渡すために、ベンダはソフトウェアを記録するためにフロッピーディスク、又はコンパクトディスクを購入し、ソフトウェアをそれらディスク上にコピーし、ディスクにラベルを貼り、適切な顧客及び担当者がだれであるかを決定し、宛名ラベルを貼り、これらのディスクを正しい個人に対して郵送しなければならない。しかし、上述のように、最新バージョンのソフトウェアを含むこれらのディスクが正しい場所に届いたとしても、そのソフトウェアが正しくインストールされる保証はどこにもない。バグ、又は不正確なインストールのいずれに起因するかに関わらずソフトウェアに関するあらゆる問題は、間違いなくそのソフトウェアを販売したソフトウェア会社の責任とされる。

【０００５】自社のソフトウェアのアップデートバージョンが、自社の顧客の１人である正しい者により適時に受け取られ、正しくインストールされることを確実にすることがソフトウェア会社の最も高い関心事項であることは明らかである。

【０００６】ソフトウェア会社が直面する他の問題は、その会社のソフトウェアを実行している間に顧客が直面したソフトウェアのバグの解明、及び修正である。通常、ソフトウェア実行中にエラーメッセージを受けた顧客は、ソフトウェア会社に連絡を取り、そして、問題を取り巻く状況を説明する。このようなユーザは、一般にその時何をしていたかを説明し、発生した問題を詳細に説明し、表示された全てのエラーＩＤ番号を報告することが出来る。しかしながら、コンピュータの構成、他の実行中プログラム、及び内部変数の状態を含む詳細事項が報告されることはまれである。本質的に、問題発生時、顧客がソフトウェア会社から離れた場所におり、エラー発生時にシステムの「スナップショット」を転送する手段を有しないことが問題である。

【０００７】当然ながら、顧客がソフトウェアの実行を中断し、問題を無視し、又は、エラーに関する情報を保有するソフトウェア会社に対する連絡を選択しない場合には、ソフトウェア会社は不利な立場に置かれる。これらの状況では、ベンダは問題の発生を知らないので、問題を修正する機会が与えられない。ソフトウェア中のエラーは、非常に複雑且つ微妙なので、問題を再現し、解決するためには、ソフトウェア会社にとって問題発生下の状況を正確に知ることが非常に重要である。遠隔地の遠隔コンピュータでソフトウェアを実行する顧客にとって、発生した問題に関する全ての情報を顧客がソフトウェア会社に送るということは考えられないことであると共に、非常に困難である。

【０００８】したがって、それにより遠隔地からソフトウェアベンダが顧客を把握し、新規バージョンを用いて顧客のソフトウェアをアップデートし、また、ソフトウェアのバグを修正することのできる技術を提供することを目的とする。

【０００９】

【課題を解決するための手段】上記目的を達成するために本発明によれば、それによりソフトウェアベンダが自社の顧客に対してインターネットを介して遠隔地からソフトウェアの技術支援を提供し得る技術が提供される。エンドユーザがベンダのソフトウェアの実行を開始すると、ダイアログボックスがユーザに対してソフトウェア

をベンダに登録する機会を与える。ユーザがソフトウェアの登録を決定する場合には、登録を達成するためにユーザのプラットホームとベンダとの間にインターネットを介して接続が確立される。また、その後、ユーザがソフトウェアの実行を開始する毎に、ソフトウェアのアップデートが必要であるか否かを決定するためにベンダに対するインターネット接続が確立される。ユーザのソフトウェアの現バージョンと比較するために、各ソフトウェアモジュールのバージョンデータと共にベンダにとって既知の全ての重大なバグデータがユーザのプラットホームにダウンロードされる。ダウンロードされたバージョンのデータがユーザのモジュールのバージョンと一致する場合、また、既知のバグが存在しない場合には、ユーザのソフトウェアは変更されない。しかしながら、ユーザのソフトウェアの１つ以上のモジュールが古く、あるいは、重大なバグに影響されていることが比較結果から明らかな場合には、それら各モジュールについてアップデートを望むか否かを尋ねるダイアログボックスがユーザに対して表示される。リストにはアップデートが要求されるモジュールが確持される。この方法で全てのモジュールが検査されると、アップグレードリスト中の最新バージョンのモジュールがベンダからダウンロードされると共に、ユーザのプラットホーム上の対応する古いモジュールに上書きされる。

【００１０】本発明の他の実施形態によれば、それによりベンダがインターネットを介して遠隔地から、自社のソフトウェア実行時におけるエラーを有効に修復し得る方法が提供される。ユーザはベンダのソフトウェア実行時にエラーに直面すると、ソフトウェアの実行を中断し、それによりエラーが報告されると共に既知の重大なバグデータがユーザのプラットホームにダウンロードされるベンダに対するインターネット接続が確立される。エラーがベンダにより既に詳細に記録されている既知のバグのいずれかに相当するか否かを決定するために、重大なバグデータはユーザのソフトウェアの状態と比較される。エラーが既知のバグに相当する場合には、ユーザには、例えば、ダイアログボックスを介してベンダから修正ソフトウェアをダウンロードするという選択肢が与えられる。これは、例えば、バグを有するモジュールを上書きするための置換モジュールであり得る。しかしながら、エラーが既知のバグのいずれにも相当しない場合には、ベンダのプラットホーム上にエラー発生下の状況を再現するために、ベンダのプラットホーム上のソフトウェアは、どのような情報がユーザのプラットホームからアップロードされなければならないか、また、ユーザの承認の対称となるかを決定する。これはエラーを遠隔的に、すなわち、ベンダのプラットホームで解決することを目的とするものである。ユーザがアップロードを承認しない場合には、ベンダはユーザに対して（可能性ある範囲に亘）エラーの性質及びユーザが問題を共に解決し得る適切な担当者に関する情報を提供する。

【００１１】したがって、本発明は第１のソフトウェアをアップグレードする方法を提供する。本発明に従えば、第１のプラットホーム上の第１のソフトウェアの実行に応じて第１のプラットホームと遠隔プラットホームとの間に接続が確立される。第１のソフトウェアに関連する第１バージョンのデータが遠隔プラットホームからの第２バージョンのデータと異なる場合には、第１のソフトウェアの第１の部分が遠隔プラットホームからの第１アップデートコードで上書きされる。

【００１２】本発明はまた、第１のソフトウェアの問題を解決するための方法を提供する。本発明に従えば、第１のプラットホーム上の第１のソフトウェアの実行エラーに応じて第１のプラットホームと遠隔プラットホームとの間に接続が確立される。エラーが遠隔プラットホームからの既知のエラーデータと一致する場合には、第１のソフトウェアの部分がアップデートコードで上書きされる。

【００１３】

【発明の実施の形態】図１はその中で本発明に係る発明の実施の形態が実行され得るネットワーク環境を簡単に示す。顧客プラットホーム１００は、そこに特定のベンダからのソフトウェアがインストールされるワークステーション、又は、パーソナルコンピュータ（ＰＣ）である。本発明と共に用いられ得る、顧客プラットホーム１００にインストールされるソフトウェアの型の例は、共通に譲渡した同時継続中の米国特許出願「自動回路設計のための方法及び装置」に記載されている。例えば、プログラマブル論理回路といった回路を設計するために用いられるソフトウェアは、非常に複雑であり、本発明の特徴から特に利益を得ることのできる型のソフトウェアである。すなわち、このようなソフトウェアは通常、顧客の満足を確実にするために、そのベンダが大きな技術支援能力を維持することを要求する。しかしながら、本明細書中で用いられる技術は、本発明の範囲を逸脱することなく幅広いソフトウェアに適用され得ることが理解される。

【００１４】顧客プラットホーム１００は、順にルータ１０８を介してインターネット１０６に接続されるファイルサーバ１０４を備えるローカルエリアネットワーク（ＬＡＮ）１０２上に存在する。ベンダプラットホーム１１０は、ＬＡＮ１０２から離れているファイルサーバであり、ルータ１１２を介してインターネット１０６に接続されている。図１に示されているネットワーク環境は、その中で本発明が実行され得る環境の一例であり、本発明の範囲を逸脱することなく幅広いネットワーク構成が採用され得る。

【００１５】本発明に従えば、顧客はインターネットを介してベンダに対するソフトウェアの登録を実行し得る。本発明のこの局面の詳細な実施の形態について、図

２のフローチャート２００を参照して説明する。顧客又はエンドユーザが自身のプラットホーム上でソフトウェアの実行を開始すると（ステップ２０２）、ソフトウェアはインターネットに対するアクセスが利用可能であるか否かを決定する（ステップ２０４）。利用不可能な場合には、ユーザに対してソフトウェアの登録についてソフトウェアベンダに連絡を取るよう指示するダイアログボックスが表示される（ステップ２０６）。一方、インターネットに対するアクセスが利用可能な場合には、ソフトウェアは、ユーザに対してソフトウェアの登録の承認を促すダイアログボックスを表示する（ステップ２０８）。ユーザが登録手続きの進行を承認しない場合には（ステップ２１０）、ソフトウェア登録ルーチンは終了し、ユーザは通常のソフトウェアの実行を続行し得る。しかしながら、ユーザが承認する場合には、それによりソフトウェアの登録が実行される（ステップ２１４）インターネット接続がユーザのプラットホームとベンダとの間に確立される（ステップ２１２）。一度、ソフトウェアが登録されると、ユーザのプラットホーム上における通常のソフトウェアの実行を続行し得る。

【００１６】上記方法で達成されるソフトウエアの登録により多くの利点がもたらされる。ここに説明する登録手続きの自動化により、この手続きを採用するソフトウェア製品の登録は、例えば、郵便による登録といった具合に、登録の実行の負担が全て顧客にかかる場合よりも行われ易いように思われる。また、ベンダは、誰が自社のソフトウェアを購入したかを知っているとき、自社の顧客が自社製品の悪癖らしい長所を使用可能にするために必要な技術支援を提供するためにより良い立場にある。さらに、適切な顧客データを用いることにより、ベンダは、自社の顧客の要求に対応するより良い製品設計戦略を展開することができる。

【００１７】図３は、本発明の詳細な発明の実施の形態に従う、それにより顧客のソフトウェアがアップデートされ得るフローチャートを示す。顧客、又は、エンドユーザが自身のプラットホーム上でソフトウェアの実行を開始すると（ステップ３０２）、ソフトウェアはインターネットに対するアクセスが利用可能であるか否かを決定する（ステップ３０４）。利用不可能であると共に、ユーザのソフトウェアが所定の時効（aging、エージング）期間よりも古い場合には、ユーザがソフトウェアの最新バージョンを有しているか否かの決定に関してソフトウェアベンダに連絡を取ることをユーザに指示するダイアログボックスが表示される（ステップ３０６）。例えば、ソフトウェアが３ヶ月に一度アップデートされる場合には、ユーザのプラットホーム上のソフトウェアのバージョンが３ヶ月以上経過している場合にダイアログボックスが表示される。

【００１８】一方、インターネットアクセスが利用可能な場合には、ソフトウェアは、この決定に関してベンダに対する接続の承認を促すダイアログボックスを表示する（ステップ３０８）。いずれの場合にも、ユーザにはまた、ソフトウェアのアップグレードに関して再度、問われることを望まないことを指示する選択肢が与えられる。ユーザが接続を承認しない場合には（ステップ３１０）、ソフトウェアアップグレードルーチンは終了し、ユーザは通常のソフトウェアの実行を続行し得る。しかしながら、ユーザが承認する場合には、それによりアップグレード手続きが実行されるインターネット接続がユーザプラットホームとベンダとの間に確立される（ステップ３１２）。

【００１９】ソフトウェアの最新バージョン、及びベンダにとって既知である任意の重大なソフトウェアのバグに関するデータは、ベンダからユーザのプラットホームに対してダウンロードされる（ステップ３１４）。これらデータは、ユーザのソフトウェアが適切にアップグレードされているか否か、及びソフトウェアのいずれかのモジュールが既知のバグの影響を受けているか否かを決定するために用いられる。ユーザのソフトウェアの各個々のモジュールは選択され、また、バージョン及び重大なバグデータについて比較される（ステップ３１６）。選択モジュールのバージョン数、及びバージョンデータが一致する場合（ステップ３１８）及び、検査されていないモジュールが残存している場合（ステップ３１９）には、比較のために他のモジュールが選択される。ステップ３１８にて選択されたモジュールが最新バージョンでないと決定された場合には、重大なバグデータが選択モジュールと一致するか否かを決定する（ステップ３２０）。一致しない場合には、ユーザに対してモジュールがアップデートされるべきか否かを尋ねるダイアログボックスが表示される（ステップ３２２）。ユーザがモジュールはアップデートされるべきであると指示する場合には、そのモジュールはアップデートリストに加えられる（ステップ３２４）。ステップ３２０にて選択モジュールが重大なバグを有すると識別された場合には、ユーザに対してモジュールがアップデートされるべきか否かを尋ねるバグの性質を説明するダイアログボックスが表示される（ステップ３２６）。繰り返すと、ユーザがアップグレードを承認する場合には、モジュールはアップデートリストに加えられる。

【００２０】本発明に係る詳細な発明の実施の形態によれば、ベンダのプラットホームからダウンロードされる重大なバグデータは、特定のユーザが既知のバグに直面しそうであるか否かを決定するために用いられ得る。例えば、ソフトウェアがプログラマブル論理回路向けのソフトウェアの場合には、特定のＰＬＤ群に関して、特定のバグだけがソフトウェアの実行に影響を与える例となり得る。ソフトウェアに関連する先の使用データを研究することにより、ユーザがその装置群を使用して作業したことがあるか否か、従って、バグが修復されるべきか

否かが決定され得る。発明の実施の形態の１つでは、ソフトウェアはそれらバグについて全く解析されない。他の発明の実施の形態では、ユーザにはバグが修復されるべきか否かを決定する選択肢が与えられる。

【００２１】一旦、全てのモジュールが検査されると（ステップ３２８）、いずれかのモジュールがアップグレードリストに載せられたか否かを判断する（ステップ３３０）。アップグレードを指示されたモジュールが存在しない場合には、ソフトウェアのアップグレードルーチンが終了し、ユーザは通常のソフトウェアの実行を続行し得る。しかしながら、１個のモジュールでもユーザによってアップグレードが要求される場合には、ユーザのソフトウェアがシャットダウンされ（ステップ３３２）、新規モジュールがベンダからユーザのプラットホームへダウンロードされ（ステップ３３４）、そして、アップグレードリスト上の対応モジュールが上書きされる（ステップ３３６）。それから、適切な実行を確実にするために新規構成ソフトウェアモジュールの診断が実行される（ステップ３３８）。一旦、診断が適切な実行を示すと、ルーチンは終了し、通常のソフトウェアの実行が続行し得る。

【００２２】前記手続きの性質に基づき、顧客に最新バージョンのベンダのソフトウェアを確実に実行させるという困難な課題が飛躍的に促進される。理解されるように、ソフトウェア、特に複雑な設計ツール、のアップグレードが数多くの理由により望まれている。例えば、ソフトウェアプログラムを顧客に対して配布する前に、ソフトウェアプログラム中の全てのバグを発見することは事実上不可能である。実際、大規模顧客による使用は、あらゆる隠れた問題を出現させるために必要な条件の全てを生み出す。したがって、一旦このような問題が出現すると、そのバグにより影響を受けるソフトウェアの部分はアップグレードされなければならない。さらに、ソフトウェアパッケージに対する改良が図られる際、ベンダの関心事は、自社の顧客が改良による長所を享受し得ることを確実にすることにある。

【００２３】しかしながら、前述のように、顧客によるソフトウェアの登録は、歴史的に当てにならない。このことは、自社の顧客に関するベンダの情報が相関的に当てにならなくなっているという点でソフトウェアのアップグレードに対して障害をもたらしてきた。登録が成し遂げられた場合であっても、雇用状況が頻繁に変化するという事実に起因して、アップグレードを達成するために顧客組織内における適任者を識別することはベンダにとってしばしば困難である。

【００２４】従来より実行されてきたソフトウェアのアップグレード方法もまた問題をはらんでいる。顧客、及び担当者の識別に関連する困難さに加えて、いくつかのディジタル記録媒体での新規コードの配布は、費用がかさむと共に時間を浪費する。さらに、一旦、新規コードが顧客の手に渡ると、新規コードが顧客のシステム上に適切にインストールされるという保証はどこにもない。実際、ソフトウェアの実行に当たり顧客が直面する多くの問題は、ソフトウェアの不適切なインストール、及び／又は、ソフトウェアの不適切な構成の結果もたらされる。これらの問題は、特に、新規ソフトウェアの引渡し直後の期間、年に数回のアップグレードが必要とされ得る事実によって悪化される。

【００２５】本明細書中に記載されるソフトウェアアップグレード方法は、これらの各問題を解決する。たとえ如何に多くのアップグレードが行われようと、アップグレードが可能であるとき、及び／又は、推奨されるとき、ユーザはブートアップする度に尋ねられるので、いわんや各ユーザはソフトウェアの最新バージョンを所有しているように思われる。さらに、アップグレードが各顧客について直接実行され、その処理に関わる個人の役割はもはや存在しないので、各顧客毎に適切な連絡者を把握し続けるという必要性が排除される。従来の配布方法の特徴である費用、及び遅延もまたもはや問題ではなくなる。出荷、生産、及び広告費用が事実上排除され、顧客によるアップグレードの承認の拒否だけが唯一遅延をもたらす。最後に、ソフトウェアのインストール及び構成が全くベンダの管理下のみにあるので、不適当なインストールの発生は劇的に低減されるはずである。さらに、（例えば、ベンダが図２を参照して説明した登録手続きを採用している場合）ベンダが自社の顧客に関する適切な記録を保有している場合には、アップグレードを望んだ顧客を把握し続けることにより特定の顧客が自社製品の最新バージョンを使用しているか否かを個々に把握することができる。単純に自社の個々の顧客によって使用されている現行バージョンを知ることにより、ベンダは非常に高い水準の顧客サービスを提供することができる。

【００２６】図４ａは、それにより遠隔プラットホームからユーザのプラットホーム上のソフトウェア上で診断ルーチンが実行される技術を説明するフローチャート４００である。ユーザのプラットホーム上でソフトウェアを実行中に（ステップ４０２）、エラーが発生し、ユーザが自身の評価としてエラーが発生した旨を示すか、または、ユーザがソフトウェアの強化を望むか（ステップ４０４）のいずれかのとき、ソフトウェアはベンダのソフトウェアに接続するためにインターネットに対するアクセスが利用可能であるか否かを決定する（ステップ４０６）。インターネットアクセスが利用不可能な場合、ダイアログボックスは、ユーザに対してベンダと連絡を取り、例えば、適切な担当者の名前、電話番号、電子メールアドレスを提供するように指示する（ステップ４０７）。しかしながら、インターネットアクセスが利用可能な場合には、ユーザに対して遠隔問題解決が開始されるべきか否かを尋ねるダイアログボックスが生成される

（ステップ４０８）。ユーザが肯定的に応じた場合には（ステップ４１０）、ユーザのプラットホームとソフトウェアベンダとの間にインターネットを介する接続が確立され（ステップ４１２）、エラーが表示される場合には（ステップ４１３）、重大なバグのテーブルがベンダのプラットホームからユーザのプラットホームへダウンロードされる（ステップ４１４）。あるいは、ルーチンは直接ステップ４３７に移行する。重大なバグのテーブルは、それにより関連する重大なバグが特徴付けられるソフトウェアの内部状態のリストである、関連状態リストを各々が有する既知のバグのリストを含んでいる。これら内部状態は、重大なバグが発生した後のソフトウェア中の内部変数の状態を表す。

【００２７】重大なバグがテーブルから選択され（ステップ４１６）、また状態が選択された重大バグに関連する状態リストから選択される（ステップ４１８）。選択状態がユーザのソフトウェアの内部状態と一致する（ステップ４２０）と共に、未だ検査されていない状態がリスト中に存在する（ステップ４２２）場合には、状態リスト中の次の状態が選択される。ルーチンがリスト中の全状態を通じて上手くやり遂げる、すなわち、リスト中の全状態がユーザのソフトウェアの内部状態と一致する場合には、関連する重大なバグがソフトウェアエラーの原因であると見なされ、例えば、ダイアログボックスを介してユーザに報告される（ステップ４２４）。しかしながら、ルーチンが状態リスト中でソフトウェアの内部状態と一致しない一つの状態と直面するやいなや、また、重大バグのテーブル中の全ての重大なバグが検査されていない場合には（ステップ４２６）、それの状態リストとソフトウェアとを比較するためにテーブル中の次のバグが選択される。

【００２８】一旦、ステップ４２４にて既知のバグがユーザに対して報告されると、ユーザは、バグを修正するためにソフトウェアをアップグレードすべきか否かを問われる（ステップ４２８）。ユーザの承認に対応して、ユーザのソフトウェアがシャットダウンされ（ステップ４３０）、また、識別されたバグに対応するソフトウェアモジュールがベンダのプラットホームからダウンロードされると共にユーザのプラットホーム上における対応モジュールが上書きされる（ステップ４３２）。その後、ソフトウェアが適切に構成されると共に実行することを確実にするために診察が実行される（ステップ４３４）。ステップ４２８でユーザがアップグレードを承認しない場合には、ユーザは上述のようにベンダに連絡を取るよう指示される（ステップ４０７）。

【００２９】ステップ４２６にて全ての重大なバグが検査されると共に、１つも一致するものが発見されない場合には、エラーは未知のバグに起因すると思われる旨をユーザに対して知らせるダイアログボックスが生成される（ステップ４３６）。そして、問題解決のためにユーザのプラットホームからベンダのプラットホーム上へ全ての必要なファイルをアップロードするために承認が要求される（ステップ４３７）。ユーザが必要なファイルのアップロードを承認する場合には（ステップ４３８）、ファイルリスト、及びユーザのシステム及びソフトウェアの状態に関する他の必要な情報がコンパイルされ、また、アップロードについての第２の通知済承認を得るためユーザに対して表示される（ステップ４４０）。ユーザがリストに目を通し、承認を与える場合には、情報はベンダのプラットホームにアップロードされ（ステップ４４４）、その確認がユーザに送り返される（ステップ４４６）。ユーザのファイル及びユーザのシステムからの他の情報を用いて、ベンダは障害が発生した状態を再現することができ、これにより問題を再現し、願わくば問題を解決する。情報のアップデートについての承認がステップ４３８又はステップ４４２のいずれかのステップにて得られない場合には、ユーザは、上述のようにベンダと連絡を取るよう指示される（ステップ４０７）。

【００３０】図４ｂは、それによりユーザのプラットホームのソフトウェア上にて診察ルーチンが遠隔的に実行される他の技術を説明するフローチャート４０１である。ステップ４０２～ステップ４３６は、図４ａを参照した上記説明と同様である。しかしながら、重大なバグのテーブル中の全ての重大なバグが検査された場合、すなわち、一致が発見されなかった（ステップ４２６）場合には、それによりエラーは未知のバグに起因すると思われる旨をユーザに対して知らせるダイアログボックスが生成される（ステップ４３６）。その後、ユーザのプラットホーム上のファイルを使用してベンダのプラットホームから遠隔的にソフトウェアを実行するために承認が要求される（ステップ４３７'）。承認が得られる場合には（ステップ４３９）、ベンダのプラットホームからソフトウェアが実行される（ステップ４１１）。承認が得られない場合には、ユーザはベンダに連絡を取るように指示される（ステップ４０７）。

【００３１】本発明に係るこの詳細な発明の実施の形態における遠隔駆動は、エラー状態に達するまでユーザのソフトウェア命令を通じて断続的に進められる。すなわち、この技術は、プログラマブル論理回路設計にあってはかなり大きい可能性のあるユーザの全ファイルをアップロードする必要なく最初にユーザが直面したエラー状態の再現を試みる。この遠隔駆動技術を通じて、ユーザのソフトウェアの特定モジュールが問題の起因であると認定される場合には（ステップ４４３）、ユーザに対して欠陥モジュールが報告される（ステップ４４５）と共に、そのモジュールは、図４ａのステップ４２８～ステップ４３４を参照して前述したようにアップデートされ得る。ユーザファイルの１つが問題の起因である認定される場合には（ステップ４４７）、認定問題の解決に関

する追加支援に関するベンダ連絡情報と共に欠陥ファイルがユーザに対して報告される（ステップ４４９）。遠隔駆動が欠陥ソフトウェアモジュール、及び欠陥ユーザファイルのいずれも認定できない場合には、ユーザは上述のようにベンダに連絡を取るように指示される（ステップ４０７）。

【００３２】本発明の上記発明の実施の形態の長所は明白である。これらの内、重要なことは、ソフトウェアプログラム実行中におけるエラーの発生、及びベンダによるエラーの診察及び修正が劇的に低減されるという事実である。２、３のダイアログボックスを除き、手続きは実質的に自動である。ユーザの問題が未知のバグに起因する場合であっても、本発明は、できる限り迅速且つ有効に問題を解決するために用意される他の水準の問題解決を提供する。各場合において、エラーが適切に再現され、これにより適切な問題解決手段が確実に得られるようにするため、ベンダはユーザのシステム及びデータファイルの適切な「スナップショット」を取得する。

【００３３】顧客の問題を素早く解決する重要さは誇張された話ではない。プログラマブル論理回路設計の市場は、この点についての例である。その間、設計過程が進行しないプログラマブル論理回路設計中のあらゆる期間は、非常に高くつく。この技術領域におけるベンダの主な関心事の１つは、チップを市場に置くために必要な時間量を削減することである。設計を促進すると共に支援する本発明が備える能率は、この目標に向かう大きな進歩である。

【００３４】図５は、ソフトウェアのユーザがソフトウェアのベンダによって管理されている遠隔プラットホーム上に存在するヘルプファイルにアクセスし得る方法を説明するフローチャート５００である。ソフトウェア実行中、ユーザがヘルプを必要とするとき（ステップ５０２）、ユーザのプラットホーム上のソフトウェアは、インターネットに対するアクセスが利用可能であるか否かを決定する（ステップ５０４）。インターネットアクセスが利用不可能な場合、ユーザのソフトウェアは、ユーザに対して如何にしてベンダから技術支援を取得するかに関する情報を提供するダイアログボックスを生成する（ステップ５０６）。しかしながら、インターネットアクセスが利用可能な場合には、ユーザに対してベンダから利用可能なオンラインヘルプについて通知するダイアログボックスを生成する（ステップ５０８）。ユーザがオンラインヘルプの使用を決断する場合には（ステップ５１０）、インターネットを介してユーザのプラットホームとベンダとの間に接続が確立される（ステップ５１２）。あるいは、ステップ５０６にてユーザに技術支援情報が提供される。その後、ヘルプトピックスのメニューがベンダのプラットホームからダウンロードされ、ユーザに提供される（ステップ５１４）。ユーザによる１つ又は複数のトピックスの選択に応じて（ステップ５１６）、選択トピックスに関連するヘルプファイルがユーザのプラットホームにダウンロードされる（ステップ５１８）。ユーザが選択しない場合には、すなわち、要求を取り消す場合には、ステップ５０６を参照して上述した支援情報が提供される。

【００３５】上記オンライン技術支援は、その中で本発明がソフトウェアプログラムの最適な実行を促進する他の方法である。ヘルプファイルは、ベンダのプラットホーム上に位置するので、ソフトウェアの現在状態を反映するために及び、時間の経過に伴いベンダにとって明らかになっている可能性のある任意の追加情報を組み込むために必要に応じてアップデートされ得る。本発明によれば、そのような情報が収集され得る１つの方法は、ベンダによって自社の顧客からのヘルプ要求を監視し続けることである。自社のヘルプユーティリティの使用に関する統計量を編集することにより、ベンダはどのヘルプファイルが最も頻繁に要求されるかが決定することができる。この情報は、順に、自社製品を改良するためにベンダの努力の方向性の手引きとして、また、技術支援リソースの焦点化に役立ち得る。

【００３６】本発明の個々の局面は全て、より迅速な水準のサービス及び技術支援を提供するためにソフトウェアベンダとその顧客との間における密接なつながりを意図する。この範例の結果、認識され得る他の多くの長所が存在する。例えば、ソフトウェアベンダは、自社の顧客が興味を持つであろう新規製品及び関連製品に関する情報を自社の顧客に対して提供するために顧客のプラットホームに対して頻繁に接続するという利点を得ることができる。さらには、一定期間かけて収集された特定の顧客に関する情報に基づき、ベンダは、詳細に各顧客の要求に合わせたフォーマット及び内容でそれらの情報を提供することができる。したがって、本発明はまた、適応性を促進すると共に、将来を見越した顧客とベンダ間の関係のために有用である。

【００３７】図６は、本発明に係る発明の実施の形態のソフトウェアを実行するために用いられ得るコンピュータシステムの例を示す。すなわち、例えば、システム６０１は、図１のワークステーション１００及び／又はサーバ１１０の例示である。コンピュータシステム６０１は、ディスプレイ６０３、スクリーン６０５、キーピキャット６０７、キーボード６０９、及びマウス６１１を備える。マウス６１１は、グラフィカルユーザインターフェースと対話するための１個以上のボタンを備えている。キャビネット６０７は、ＣＤ－ＲＯＭドライブ６１３、本発明と共に用いるデータ等を実行するコンピュータコードを組み込むソフトウェアプログラムを格納すると共に引き出すために用いられ得るシステムメモリ、及びハードドライブ（図７参照）を収容する。ＣＤ－ＲＯＭ６１５がコンピュータ読取可能媒体の例示として図示されているが、フロッピーディスク、テープ、フラッシ

ュメモリ、システムメモリ、及びハードドライブを含む他のコンピュータ読取可能媒体が用いられ得る。

【００３８】図７は本発明に係る発明の実施の形態のソフトウェアを実行するために用いられるコンピュータシステム６０１のシステムブロック図を示す。図６に示すようにコンピュータシステム６０１は、モニタ６０３及びキーボード６０９、及びマウス６１１を備える。コンピュータシステム６０１はさらに、中央演算処理装置７０１、システムメモリ７０３、固定ディスク７０５（例えば、ハードドライブ）、リムーバブルディスク７０７（例えば、ＣＤ－ＲＯＭドライブ）、ディスプレイアダプタ７０９、サウンドカード７１１、スピーカ７１３、及びネットワークインターフェース７１５といったサブシステムを備えている。本発明と共に用いることが適切な他のコンピュータシステムは、追加の、又は、少ないサブシステムを備え得る。例えば、他のコンピュータシステムは、１個以上の演算処理装置（すなわち、マルチプロセッサシステム）、あるいは、キャッシュメモリを備えることができる。

【００３９】コンピュータシステム６０１のシステムバスアーキテクチャは、矢印７１７にて表されている。しかしながら、これらの矢印は、サブシステムをリンクするために機能する任意の内部接続方式を意味する。例えば、ローカルバスは、中央演算処理装置をシステムメモリ及びディスプレイアダプタに接続するために用いられ得る。図７に示すコンピュータシステム６０１は、本発明と共に用いることが適切であるコンピュータシステムの例にすぎない。異なるサブシステム構成を備える他のコンピュータアーキテクチャもまた用いられ得る。

【００４０】以上、いくつかの発明の実施の形態に基づき本発明を説明したが、本発明の趣旨、あるいは、範囲を逸脱することなく当業者により種々の変更改良なされ得ることは理解されるべきである。例えば、本明細書中のいくつかの発明の実施の形態は、プログラマブル論理回路を設計するためのソフトウェアを参照して説明された。しかしながら、本明細書中に述べた原理は幅広いソフトウェアに適用可能であり得る。したがって、本発明の範囲は特許請求の範囲に基づき決定されるべきである。

【図面の簡単な説明】

【図１】　本発明の実施の形態にて実行され得るネットワーク環境の説明図である。

【図２】　本発明の実施の形態に従うソフトウェアの遠隔登録を説明するフローチャートである。

【図３】　本発明の実施の形態に従うソフトウェアの遠隔アップデートを説明するフローチャートである。

【図４】　本発明の実施の形態に従うソフトウェアの遠隔修正を説明するフローチャートである。（図４には、外国語図面の図４ａの翻訳文が示されている。）

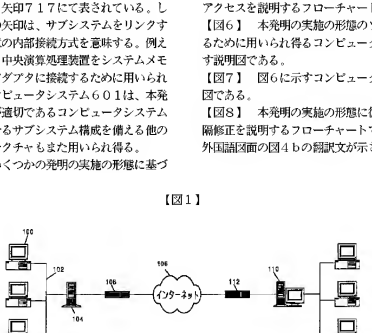
【図５】　本発明の実施の形態に従うヘルプファイルのアクセスを説明するフローチャートである。

【図６】　本発明の実施の形態のソフトウェアを実行するために用いられ得るコンピュータシステムの一例を示す説明図である。

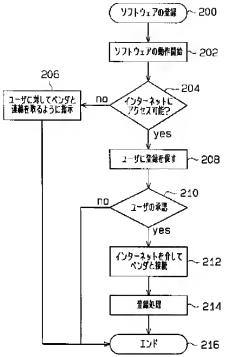
【図７】　図６に示すコンピュータシステムのブロック図である。

【図８】　本発明の実施の形態に従うソフトウェアの遠隔修正を説明するフローチャートである。（図８には、外国語図面の図４ｂの翻訳文が示されている。）

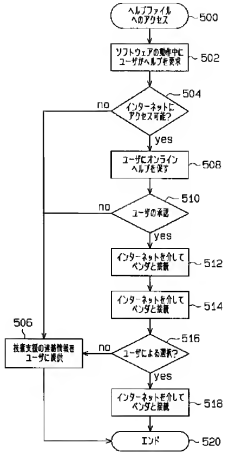
【図１】



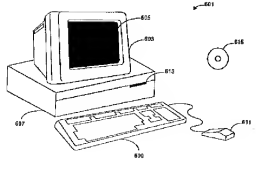
【図2】


ソフトウェアの登録
200
ソフトウェアの動作開始
202
206
ユーザに couldてベンダと連絡を取るように指示
no
インターネットにアクセス可能?
204
yes
ユーザに登録を促す
208
no
ユーザの承認
210
yes
インターネットを介してベンダと接続
212
登録処理
214
エンド
216

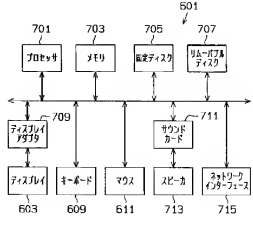
【図5】


ヘルプファイルへのアクセス
500
ソフトウェアの動作中にユーザがヘルプを要求
502
no
インターネットにアクセス可能?
504
yes
ユーザにオンラインヘルプを促す
508
no
ユーザの承認
510
yes
インターネットを介してベンダと接続
512
インターネットを介してベンダと接続
514
506
技術支援の連絡情報をユーザに提示
no
ユーザによる選択?
516
yes
インターネットを介してベンダと接続
518
エンド
520

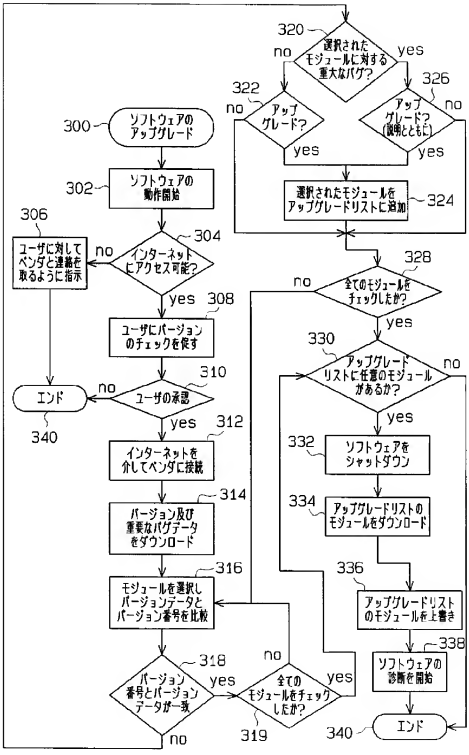
【図6】


601
605
607
612
616
611
630

【図7】

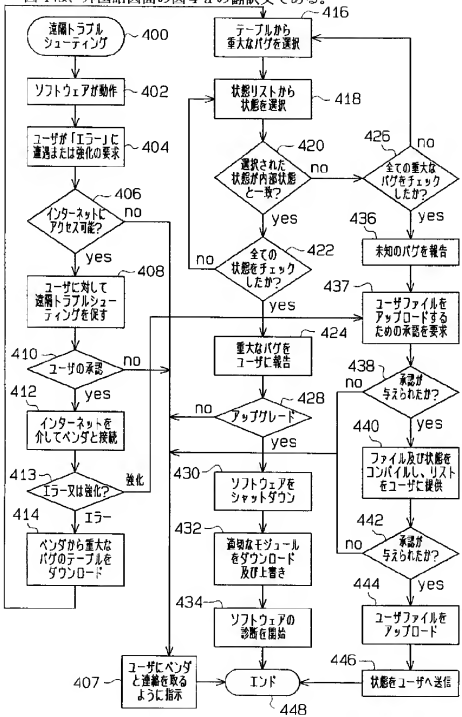

501
701
703
705
707
プロセッサ
メモリ
固定ディスク
リムーバブルディスク
709
711
ディスプレイアダプタ
サウンドカード
ディスプレイ
キーボード
マウス
スピーカ
ネットワークインターフェース
603
609
611
713
715

【図3】


ソフトウェアのアップグレード
300
ソフトウェアの動作開始
302
インターネットにアクセス可能?
304
no
ユーザに対してベンダと連絡を取るように指示
306
yes
ユーザにバージョンのチェックを促す
308
ユーザの承認
310
no
エンド
340
yes
インターネットを介してベンダに接続
312
バージョン及び重要なバグデータをダウンロード
314
モジュールを選択しバージョンデータとバージョン番号を比較
316
バージョン番号とバージョンデータが一致
318
yes
no
全てのモジュールをチェックしたか?
319
no
yes
選択されたモジュールに対する重大なバグ?
320
yes
no
アップグレード?
322
yes
選択されたモジュールをアップグレードリストに追加
324
アップグレード?(調停とともに)
326
yes
no
全てのモジュールをチェックしたか?
328
no
yes
アップグレードリストに任意のモジュールがあるか?
330
yes
no
ソフトウェアをシャットダウン
332
アップグレードリストのモジュールをダウンロード
334
アップグレードリストのモジュールを上書き
336
ソフトウェアの診断を開始
338
エンド
340

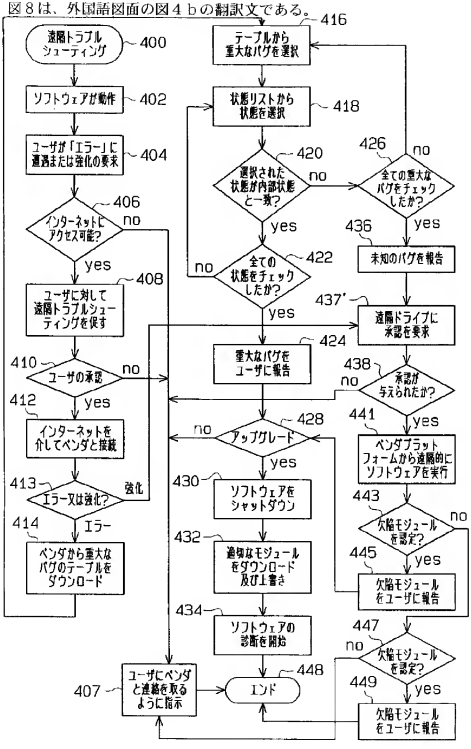
【図4】

図４は、外国語図面の図４ａの翻訳文である。



【図8】

図８は、外国語図面の図４ｂの翻訳文である。



【外国語明細書】

METHOD FOR PROVIDING REMOTE SOFTWARE TECHNICAL SUPPORT

RELATED APPLICATION DATA

The present application claims priority from U.S. Provisional Application Serial No. 60/029,277 entitled TOOLS FOR DESIGNING PROGRAMMABLE LOGIC DEVICES filed on October 28, 1996, the entire specification of which is incorporated herein by reference.

This invention is related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP017/A343), filed on the same day as this patent application, naming B. Pedersen et al. as inventors, and entitled "GENERATION OF SUB-NET LISTS FOR USE IN INCREMENTAL COMPILATION." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP019/A345), filed on the same day as this patent application, naming J. Tse et al. as inventors, and entitled "FITTING FOR INCREMENTAL COMPILATION OF ELECTRONIC DESIGNS." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP033/A404), filed on the same day as this patent application, naming D. Mendel as inventor, and entitled "PARALLEL PROCESSING FOR COMPUTER ASSISTED DESIGN OF ELECTRONIC DEVICES." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP008/A334), filed on the same day as this patent application, naming F. Heile et al. as inventors, and entitled

"INTERFACE FOR COMPILING DESIGN VARIATIONS IN ELECTRONIC DESIGN ENVIRONMENTS." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP006/A318), filed on the same day as this patent application, naming T. Southgate as inventor, and entitled "METHOD AND APPARATUS FOR AUTOMATED CIRCUIT DESIGN." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP013/A339), filed on the same day as this patent application, naming T. Southgate et al. as inventors, and entitled "GRAPHIC EDITOR FOR BLOCK DIAGRAM LEVEL DESIGN OF CIRCUITS." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP014/A340), filed on the same day as this patent application, naming T. Southgate et al. as inventors, and entitled "DESIGN FILE TEMPLATES FOR IMPLEMENTATION OF LOGIC DESIGNS." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP016/A342), filed on the same day as this patent application, naming T. Southgate as inventor, and entitled "METHOD FOR PROVIDING REMOTE SOFTWARE TECHNICAL SUPPORT." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP018/A344), filed on the same day as this patent application, naming T. Southgate as inventor, and entitled "METHOD FOR SIMULATING A CIRCUIT DESIGN" That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP007/A333), filed on the same day as this patent application, naming F. Heile et al. as inventors, and entitled "WORKGROUP COMPUTING FOR ELECTRONIC DESIGN AUTOMATION." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP012/A338), filed on the same day as this patent application, naming F. Heile as inventor, and entitled "LOCAL COMPILATION IN CONTEXT WITHIN A DESIGN HIERARCHY." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP010/A336), filed on the same day as this patent application, naming Alan L. Herrmann et al. as inventors, and entitled "EMBEDDED LOGIC ANALYZER FOR A PROGRAMMABLE LOGIC DEVICE." That application is incorporated herein by reference in its entirety and for all purposes.

This invention is also related to U.S. Patent Application Serial No. 08/___,___ (attorney docket no. ALTRP015/A341), filed on the same day as this patent application, naming F. Heile as inventor, and entitled "ELECTRONIC DESIGN AUTOMATION TOOL FOR DISPLAY OF DESIGN PROFILE." That application is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

The present invention relates to a method for providing software technical support from a remote location via, for example, the Internet. More specifically, the present invention provides a method by which software upgrades and fixes for software bugs may be incorporated into a customer's software from the remote location.

Once a software producing company has sold a software product to its customer, it is extremely important to

keep track of these customers and to update the software periodically. No software program is ever bug free; in fact, the software industry is quite accustomed to re-releasing newer versions of software programs on a periodic basis in order to correct errors in the software. These later releases may add new features to the software, may correct minor problems, or often correct critical errors in the software. Because the performance of a software program is important for customers, it is in a software company's best interest to release these later versions of software to its customers and to insure that the newer software versions are received and installed correctly.

However, numerous obstacles are present that thwart even the best efforts of a conscientious software company to periodically update its software products with newer versions for all of its customers. One problem is that few customers who buy a software program go through the formalities of registering the program and sending the registration back to the software company. A registration form allows a software company to keep track of purchasers of its software, their addresses, contact persons, the version purchased, etc. However, few such software registration forms are received by software vendors. This makes it extremely difficult for a software company to determine to whom regular updates should be sent. For those software companies that update software regularly (such as quarterly) or may need to correct critical software bugs, this lack of software registration presents a real problem. And even if a registration form has been received, it is often difficult to figure out exactly to whom the new software version should be directed. Individuals change roles within an organization, organizations merge, engineers leave, etc., all making it extremely difficult to figure out to whom within an organization a new software version should be sent. It is also important that new versions of software be sent to a responsible person within the organization to ensure that all computers running the software are updated with the new version, and that the installation is done correctly. However, even with such

competent individuals, errors in installation.

Another problem with releasing new versions of software to customers is the problem of distribution. In order to release a new version of the software, the vendor must purchase floppy or compact disks to hold the software, copy the software onto the disks, label the disks, determine who the appropriate customers and contacts are, and label and mail these disks to the correct individual. But as mentioned above, even if these disks containing the latest software version arrive at the correct location, there is no guarantee that the software will be installed correctly. Any problems encountered with the software, either due to bugs or incorrect installation will undoubtedly be blamed upon the software company that has sold the software program.

It is clearly in the best interests of a software company to ensure that updated versions of its software are timely received by the correct individuals within one of its customers, and correctly installed.

Another problem encountered by software companies is the identification and correction of software bugs encountered by customers in the course of using their software. Typically, a customer receiving an error message while using software will call the software company and relate the circumstances surrounding the problem. Such a user may be able to relate generally what he was doing at the time, describe the problem that occurred, and report any error ID number that was produced. However, it is rare that specific details involving computer configuration, other programs running, and the status of internal variables are reported. Essentially, the problem is that the customer is in a location isolated from the software company when the problem occurs and has no means of transferring a "snapshot" of his system at the time the error occurs.

Of course, if the customer discontinues use of the software, ignores the problem, or otherwise chooses not to contact the software company with news of the error, the software company is put at a disadvantage. In these situations, because the vendor does not hear about the

occurrence of the problem, it is not given the opportunity to correct it. Because errors in software can be extremely complex and subtle, it is very important for a software company to know exactly the circumstances under which the problem occurred in order to be able to reproduce the problem and to solve it. For customers using the software at a remote location on a remote computer, it is unlikely and extremely difficult that a customer will be able to transfer all of the information concerning such a problem back to the software company.

It is therefore desirable that a technique be provided by which a software vendor can track customers, upgrade customer software with newer versions, and correct software bugs from remote locations.

SUMMARY OF THE INVENTION

According to the present invention, techniques are provided by which a software vendor may provide software technical support to its customers from a remote location via the Internet. When the end user begins operating the vendor's software, dialog boxes give the user the opportunity to register the software with the vendor. If the user decides to register the software, a connection between the user's platform and the vendor is established via the Internet to effect registration. Moreover, each time thereafter that the user begins operating the software, an Internet connection to the vendor is established for the purpose of determining whether a software upgrade is necessary. Version data for each of the software's modules as well as any critical bug data known by the vendor are downloaded to the user's platform for comparison to the user's current version of the software. Where the downloaded version data agree with the user's module versions, and where there are no known bugs, the user's software remains unchanged. However, if the comparison reveals that one or more of the user's software modules are outdated or infected with a critical bug, dialog boxes are presented to the user for each such module requesting whether an upgrade is desired. A list is maintained of the modules

for which an upgrade is requested. When all of the modules have been checked in this manner, the latest version of the modules in the upgrade list are downloaded from the vendor and written over the corresponding outdated modules on the user's platform.

According to another embodiment of the invention, a method is provided by which a vendor may effectively troubleshoot errors in the operation of its software from a remote location via the Internet. When a user encounters an error in the operation of the vendor's software, operation of the software is suspended and an Internet connection to the vendor is established by which the error is reported and known critical bug data are downloaded to the user's platform. The critical bug data are compared to the state of the user's software to determine whether the error corresponds to any known bugs previously documented by the vendor. Where the error corresponds to a known bug, the user is given the option, e.g., via a dialog box, to download a software fix from the vendor. This may be, for example, a replacement module to overwrite the module with the bug. Where, however, the error does not correspond to any known bug, software on the vendor's platform determines what information must be uploaded from the user's platform, subject to the user's authorization, in order to recreate on the vendor's platform the conditions under which the error occurred; this for the purpose of troubleshooting the error locally, i.e., on the vendor's platform. If the user does not authorize the upload, the vendor supplies information to the user regarding the nature of the error (to the extent possible) and an appropriate contact person with whom the user may attempt to solve the problem.

Thus, the present invention provides a method for upgrading first software. According to the invention, in response to operation of the first software on a first platform, a connection is established between the first platform and a remote platform. Where first version data associated with the first software are different from second version data from the remote platform, a portion of the first

software is overwritten with first updated code from the remote platform.

The present invention also provides a method for troubleshooting first software. According to the invention, in response to an error in operation of the first software on a first platform, a connection is established between the first platform and a remote platform. Where the error corresponds to known error data from the remote platform, a portion of the first software is overwritten with updated code.

A further understanding of the nature and advantages of the present invention may be realized by reference to the remaining portions of the specification and the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a diagram of a network environment in which specific embodiments of the present invention may be implemented;

Fig. 2 is a flowchart illustrating remote registration of software according to a specific embodiment of the invention;

Fig. 3 is a flowchart illustrating remote upgrading of software according to a specific embodiment of the invention;

Figs. 4a and 4b are flowcharts illustrating remote troubleshooting of software according to specific embodiments of the invention;

Fig. 5 is a flowchart illustrating access of help files according to a specific embodiment of the invention;

Fig. 6 illustrates an example of a computer system that may be used to execute the software of an embodiment of the invention; and

Fig. 7 shows a system block diagram the computer system of Fig. 6.

DESCRIPTION OF SPECIFIC EMBODIMENTS

Fig. 1 is a simplified representation of a network environment in which specific embodiments of the present invention may be implemented. Customer platform 100 is a workstation or personal computer (PC) on which software from a particular vendor is installed. An example of the type of software installed on customer platform 100 which may be employed with the present invention is described in commonly assigned, copending U.S. Patent Application Serial No. 08/_____ for METHOD AND APPARATUS FOR AUTOMATED CIRCUIT DESIGN, filed simultaneously herewith, the entire specification of which is incorporated herein by reference. Software used to design circuits such as, for example, programmable logic devices, is extremely complex and, as such, is the type of software which could especially benefit from the features of the present invention. That is, such software typically requires its vendor to maintain a large technical support capability to ensure customer satisfaction. It will be understood, however, that the techniques described herein may be applied to a wide variety of software without departing from the scope of the invention.

Customer platform 100 resides on a local area network (LAN) 102 with a file server 104 which is, in turn, connected to the Internet 106 via a router 108. Vendor platform 110 is a file server which is remote from LAN 102 and is connected to Internet 106 via router 112. It will be understood that the network environment illustrated in Fig. 1 is one example of an environment in which the present invention may be practiced, and that a wide variety of network configurations may be employed without departing from the scope of the invention.

According to the invention, a customer may effect registration of software with the vendor via the Internet. A specific embodiment of this aspect of the invention will now be described with reference to flowchart 200 of Fig. 2. When the customer or end user begins operating the software on her platform (step 202), the software determines whether access to the Internet is available (step 204). If not, a dialog box is

presented instructing the user to contact the software vendor for registering her copy of the software (step 206). If, on the other hand, Internet access is available, the software presents a dialog box prompting the user to authorize registration of the software (step 208). If the user does not authorize moving forward with the registration process (step 210), the software registration routine ends and the user may proceed with normal operation of the software. However, if the user does give authorization, an Internet connection is established between the user's platform and the vendor (step 212) by which registration of the software is effected (step 214). Once the software is registered, normal operation of the software on the user's platform may proceed.

Several advantages result from software registration achieved in the above-described manner. Because of the automated nature of the registration process described herein, the registration of a software product employing this process is much more likely to occur than if the onus for effecting registration is placed entirely on the customer as in the case, for example, of registration by mail. And, when vendors know who has purchased their software, they are in a better position to provide the technical support necessary to enable their customers to use their products to their greatest advantage. In addition, with accurate customer data, vendors are better able to develop product design strategies which are responsive to the needs of their customer base.

Fig. 3 is a flowchart 300 illustrating a method by which a customer's software may be upgraded according to a specific embodiment of the invention. When the customer or end user begins operating the software on her platform (step 302), the software determines whether access to the Internet is available (step 304). If not, and the user's software is older than some predetermined aging period, a dialog box is presented instructing the user to contact the software vendor for determining whether she has the latest version of the software (step 306). If, for example, the software is updated quarterly, the dialog box is presented if the version of the software on the user's platform is more than three months old.

If, on the other hand, Internet access is available, the software presents a dialog box prompting the user to authorize connecting to the vendor for this determination (step 308). In either case, the user is also given the option of indicating that she does not wish to be asked again regarding updating the software. If the user does not authorize the connection (step 310), the software upgrade routine ends and the user may proceed with normal operation of the software. However, if the user does give authorization, an Internet connection is established between the user's platform and the vendor (step 312) by which the upgrade process may proceed.

Data are downloaded from the vendor to the user's platform regarding the most current version of the software and any critical software bugs known to the vendor (step 314). These data are used to determine whether the user's software is appropriately updated and whether any of the modules of the software are affected by the known bugs. Each individual module of the user's software is selected and compared to the version and critical bug data (step 316). Where the version number of the selected module and the version data agree (step 318) and where modules remain which have not been checked (step 319), another module is selected for comparison. If at step 318 the selected module is determined not to be the latest version, it is then determined whether the critical bug data identify the selected module (step 320). If not, a dialog box is generated asking the user whether the module should be upgraded (step 322). If the user indicates that the module should be upgraded, it is added to an upgrade list (step 324). If at step 320 the selected module is identified as having a critical bug, a dialog box describing the nature of the bug is generated asking the user whether the module should be upgraded (step 326). Again, if the user authorizes the upgrade, the module is added to the upgrade list.

According to a specific embodiment of the invention, the critical bug data downloaded from the vendor's platform may be employed to determine whether a particular user is likely to encounter a known bug. For example, where the

software is for designing programmable logic devices, it could be the case that a particular bug only affects operation of the software with respect to a particular family of PLDs. By studying previous use data associated with the software, it may be determined whether the user has ever worked with that device family and, therefore, whether the bug should be fixed. In one embodiment, the software is simply not analyzed with respect to such a bug. In another embodiment, the user is given the option of deciding whether the bug should be fixed.

Once all of the modules have been checked (step 328) it is determined whether any modules have been placed on the upgrade list (step 330). Where no modules have been designated for upgrade, the upgrade software routine ends and the user may proceed with normal operation of the software. If, however, an upgrade has been requested by the user for at least one module, the user's software is shut down (step 332) and new modules are downloaded from the vendor to the user's platform (step 334) and written over the corresponding modules on the upgrade list (step 336). A diagnostic of the newly configured software modules is then performed to ensure proper operation (step 338). Once the diagnostic indicates proper operation, the routine ends and normal operation of the software may proceed.

Due to the nature of the above-described procedure, the difficult task of ensuring that customers are using the most recent version of a vendor's software is greatly facilitated. As will be understood, the upgrading of software, especially complex design tools, is desirable for a number of reasons. For example, it is virtually impossible to discover all of the bugs in a software program before distributing it to customers. In fact, it is the use by a large customer base which produces all of the conditions needed to expose any hidden problems. Therefore, once such problems are exposed, portions of the software affected by the bugs must be upgraded. In addition, as improvements to a software package are developed, it is in a vendor's interest to make sure that its customers can take advantage of the improvements.

However, as discussed above, registration of software by a customer base has historically been inconsistent. This has provided an obstacle to upgrading software packages in that the vendor's information about its customer base has been correlatively inconsistent. Even where registration is accomplished, it is often difficult for a vendor to identify the appropriate individual in a customer's organization to effect the upgrade due to the fact that people frequently change their employment situations.

The manner in which software upgrades have been traditionally effected has also been problematic. In addition to the difficulties associated with identifying customers and contacts, the distribution of the new code on some digital storage medium is both costly and time consuming. Moreover, once the new code is in the hands of the customer, there is no guarantee that it will be installed correctly on the customer's system. In fact, many of the problems that customers encounter in the use of software is a direct result of incorrect installation and/or configuring of the software. These problems are exacerbated by the fact that, especially during the period immediately following release of new software, several upgrades a year may be necessary.

The software upgrade method described herein addresses each of these problems. Because the user is asked each time she boots up when an upgrade is available and/or recommended, it is much more likely that each user will have the latest version of the software, no matter how many upgrades occur. Moreover, the necessity for keeping track of the appropriate contact person at each customer is eliminated in that, because the upgrade is done directly with each user, there is no longer a role for such an individual in the transaction. The cost and delay which characterized previous distribution schemes is also no longer an issue. Shipping, production, and advertising costs are virtually eliminated, and the only delay is due to the customer's refusal to authorize an upgrade. Finally, because installation and configuration of the software is exclusively under the control of the vendor, the incidence of incorrect installations should

be dramatically reduced. In addition, if a vendor has accurate records regarding its customer base (as would be the case, for example, where a vendor employs the registration process described above with reference to Fig. 2), it could independently track whether or not specific customers are using the most current version of their product by keeping track of the customers who have requested an upgrade. Simply by being aware of the current version being used by its various customers, a vendor will be able to provide a greater level of customer service.

Fig. 4a is a flowchart 400 illustrating a technique by which a diagnostic routine may be performed on software on a user's platform from a remote platform. When, during operation of software on a user's platform (step 402), either an error occurs, the user indicates that in her opinion an error has occurred, or the user wishes to requests an enhancement to the software (step 404), the software determines whether access to the Internet is available (step 406) for the purpose of connecting to the vendor of the software. If Internet access is not available, a dialog box instructs the user to contact the vendor, providing, for example, the name, phone number, and e-mail address of an appropriate contact person (step 407). If, however, Internet access is available, a dialog box is generated which queries the user as to whether remote troubleshooting should begin (step 408). If the user responds affirmatively (step 410), a connection between the user's platform and the software vendor is established via the Internet (step 412) and, if an error is indicated (step 413), a critical bug table is downloaded from the vendor's platform to the user's (step 414). Otherwise, the routine proceeds directly to step 437. The critical bug table contains a list of known bugs, each of which has an associated condition list which is a list of internal conditions of the software by which the associated critical bug is characterized. These internal conditions represent the state of internal variables in the software after the critical bug has occurred.

A critical bug is selected from the table (step 416)

and a condition is selected from the condition list associated with the selected critical bug (step 418). If the selected condition matches an internal condition of the user's software (step 420) and there are conditions in the list which have not been checked (step 422), the next condition in the condition list is selected. If, the routine makes it through all of the conditions in the list, i.e., all of the conditions in the list match the internal conditions of the user's software, the associated critical bug is assumed to be the cause of the software error and is reported as such to the user via a dialog box (step 424). However, as soon as the routine encounters a single condition in the condition list which does not match the internal state of the software, and where all of the critical bugs in the critical bug table have not been checked (step 426), the next bug in the table is selected for comparison of its condition list to the software.

Once a known bug is reported to the user at step 424, the user is asked whether the software should be upgraded to correct the bug (step 428). In response to the user's authorization, the user's software is shut down (step 430) and software modules corresponding to the identified bug are downloaded from the vendor's platform and written over the corresponding modules on the user's platform (step 432). A diagnostic is then run to ensure that the software is correctly configured and operational (step 434). If at step 428 the user does not authorize the upgrade, she is instructed to contact the vendor as described above (step 407).

Where at step 426 all critical bugs have been checked and no matches found, a dialog box is generated by which the user is informed that the error is likely due to an unknown bug (step 436). Authorization is then requested to upload any necessary files from the user's platform for troubleshooting of the problem on the vendor's platform (step 437). If the user grants authorization to upload the requested files (step 438), a list of files and other necessary information regarding the user's system and the state of the software is compiled and presented to the user to obtain a second, informed authorization for the upload (step

440). Where the user has reviewed the list and given authorization, the information is uploaded to the vendor's platform (step 444), confirmation of which is transmitted back to the user (step 446). With the user's files and the other information from the user's systems, the vendor may then recreate the conditions under which the failure occurred, thereby reproducing and, hopefully, correcting the problem. If authorization for the upload of information is not granted in either of steps 438 or 442, the user is instructed to contact the vendor as described above (step 407).

Fig. 4b is a flowchart 401 illustrating another technique by which a diagnostic routine may be performed remotely on software on a user's platform. Steps 402 through 436 are the same as described above with reference to Fig. 4a. However, if all critical bugs in the critical bug table have been checked, i.e., no matches have been found (step 426), a dialog box is generated by which the user is informed that the error is likely due to an unknown bug (step 436). Authorization is then requested to remotely run the software from the vendor's platform using files on the user's platform (step 437'). If authorization is granted (step 439), the software is run from the vendor's platform (step 441). If not, the user is instructed to contact the vendor (step 407).

The remote drive of this specific embodiment of the invention steps sequentially through the software commands of the user's software until an error condition is reached. That is, this technique attempts to recreate the error condition originally encountered by the user without having to upload all of the user's files which, for a programmable logic design, for example, can be rather large. If, through this remote drive technique, a particular module of the user's software is identified as causing the problem (step 443), the defective module is reported to the user (step 445) and the module may be upgraded as described above with reference to steps 428-434 of Fig. 4a. If, one of the user's files is identified as causing the problem (step 447), the defective file is reported to the user along with vendor contact information for additional support in solving the identified

problem (step 449). If the remote drive can neither identify defective software modules nor defective user files, the user is instructed to contact the vendor as described above (step 407).

The advantages of the above-described embodiments of the invention are manifest. Prominent among these is the fact that the time between the occurrence of an error in the operation of a software program and the diagnosis and correction of the error by the vendor is dramatically reduced. The user does not have to contact the vendor. Except for a few dialog boxes, the process is virtually automatic. Even in the case where the user's problem is the result of some unknown bug, the present invention provides alternative levels of troubleshooting designed to solve the problem as quickly and efficiently as possible. In each case, the vendor gets an accurate "snapshot" of the user's system and data files so that the error is correctly reproduced, thereby ensuring that the appropriate troubleshooting approach is taken.

The importance of solving customer problems so quickly cannot be overstated. The programmable logic design market is exemplary in this regard. Any periods of time in the design of a programmable logic device during which the design process is not moving forward is extremely costly. One of the main focuses for many vendors in this area of technology is reducing the amount of time required to get a chip to market. The efficiency with which the present invention facilitates and supports the design process is a significant advancement toward this goal.

Fig. 5 is a flowchart 500 which illustrates the manner in which a user of software may access help files which are resident on a remote platform maintained by the vendor of the software. When, during operation of the software, the user requests help (step 502), the software on the user's platform determines whether access to the Internet is available (step 504). If Internet access is not available, the user's software generates a dialog box which provides information to the user regarding how to obtain technical support from the vendor (step 506). If, however, Internet

access is available, a dialog box is generated which informs the user about online help available from the vendor (step 508). If the user decides to use the online help (step 510), a connection between the user's platform and the vendor is established via the Internet (step 512). Otherwise, the user is provided with the technical support information at step 506. A menu of help topics is then downloaded from the vendor's platform and provided to the user (step 514). In response to the selection of a topic or topics by the user (step 516), help files associated with the selected topic(s) are downloaded to the user's platform (step 518). If the user does not make a selection, i.e., cancels the request, the support information described above with reference to step 506 is provided.

The online technical support described above is another way in which the present invention facilitates optimal use of a software program. Because the help files are located on the vendor's platform, they can be updated as necessary to reflect the current state of the software and to incorporate any additional information which may become apparent to the vendor over time. According to the invention, one way in which such information may be collected is through monitoring by the vendor of the help requests from its customer base. By compiling statistics regarding the use of its help utility, the vendor can determine which help files are most frequently requested. This information, in turn, may be useful in guiding the vendor's efforts to improve its product as well as in focusing technical support resources.

The various aspects of the present invention all envision a greater connectivity between a software vendor and its customers to provide a more responsive level of service and technical support. There are a number of other advantages which may also be realized as a result of this paradigm. For example, software vendors may take advantage of the frequent connections to customer platforms to provide information to their customer base regarding new and related products for which the customer might have an interest. More specifically, based on information gathered about specific customers over a

period of time, a vendor may be able to provide such information in a format and with content specifically tailored to the needs of each customer. Thus, the present invention is also useful for facilitating an adaptive and forward-looking customer-vendor relationship.

Fig. 6 illustrates an example of a computer system that may be used to execute the software of an embodiment of the invention. That is, system 601 is exemplary of, for example, workstation 100 and/or server 110 of Fig. 1. Computer system 601 includes a display 603, screen 605, cabinet 607, keyboard 609, and mouse 611. Mouse 611 may have one or more buttons for interacting with a graphical user interface. Cabinet 607 houses a CD-ROM drive 613, system memory and a hard drive (see Fig. 7) which may be utilized to store and retrieve software programs incorporating computer code that implements the invention, data for use with the invention, and the like. Although the CD-ROM 615 is shown as an exemplary computer readable storage medium, other computer readable storage media including floppy disks, tape, flash memory, system memory, and hard drives may be utilized.

Fig. 7 shows a system block diagram of computer system 601 used to execute the software of an embodiment of the invention. As in Fig. 6, computer system 601 includes monitor 603 and keyboard 609, and mouse 611. Computer system 601 further includes subsystems such as a central processor 701, system memory 703, fixed disk 705 (e.g., hard drive), removable disk 707 (e.g., CD-ROM drive), display adapter 709, sound card 711, speakers 713, and network interface 715. Other computer systems suitable for use with the invention may include additional or fewer subsystems. For example, another computer system could include more than one processor 701 (i.e., a multi-processor system), or a cache memory.

The system bus architecture of computer system 601 is represented by arrows 717. However, these arrows are illustrative of any interconnection scheme serving to link the subsystems. For example, a local bus could be utilized to connect the central processor to the system memory and display adapter. Computer system 601 shown in Fig. 7 is but an

example of a computer system suitable for use with the invention. Other computer architectures having different configurations of subsystems may also be utilized.

While the invention has been particularly shown and described with reference to specific embodiments thereof, it will be understood by those skilled in the art that changes in the form and details of the disclosed embodiments may be made without departing from the spirit or scope of the invention. For example, several of the examples in this specification were related with reference to software for designing programmable logic devices. However, it is clear that the principles described herein may be applied to a wide variety of software programs. Therefore, the scope of the invention should be determined with reference to the appended claims.

WHAT IS CLAIMED IS:

1. A method for upgrading first software comprising:

in response to operation of the first software on a first platform, establishing a connection between the first platform and a remote platform; and

where first version data associated with the first software are different from second version data from the remote platform, overwriting a first portion of the first software with first updated code from the remote platform.

2. The method of claim 1 wherein the first software comprises a plurality of software modules, and wherein the overwriting occurs for each of the plurality of software modules for which the first and second version data are different.

3. The method of claim 1 further comprising, where known error data from the remote platform correspond to the first software, overwriting a second portion of the first software with second updated code.

4. The method of claim 3 further comprising downloading the known error data from the remote platform to the first platform.

5. The method of claim 1 wherein establishing the connection comprises establishing an Internet connection.

6. The method of claim 5 further comprising determining whether access to the Internet is available.

7. The method of claim 1 further comprising obtaining authorization from a user of the first software to establish the connection.

8. The method of claim 7 further comprising, where

authorization is not obtained, providing software support information to the user of the first software.

9. The method of claim 1 further comprising shutting down operation of the first software before overwriting the first portion of the first software.

10. The method of claim 1 further comprising, after overwriting the first portion of the first software, running a software diagnostic routine on the first software.

11. The method of claim 1 further comprising downloading the first version data from the remote platform to the first platform.

12. At least one computer readable medium containing program instructions for upgrading first software, said at least one computer readable medium comprising:

computer readable code for, in response to operation of the first software on a first platform, establishing a connection between the first platform and a remote platform; and

computer readable code for, where first version data associated with the first software are different from second version data from the remote platform, overwriting a first portion of the first software with first updated code from the remote platform.

13. A computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by at least one processor, cause said at least one processor to upgrade first software by:

in response to operation of the first software on a first platform, establishing a connection between the first platform and a remote platform; and

where first version data associated with the first software are different from second version data from the remote platform, overwriting a first portion of the first

software with first updated code from the remote platform.

14. A computer readable medium containing program instructions corresponding to first software, the first software having been upgraded according to a method comprising:

in response to operation of the first software on a first platform, establishing a connection between the first platform and a remote platform; and

where first version data associated with the first software are different from second version data from the remote platform, overwriting a first portion of the first software with first updated code from the remote platform.

15. A method for troubleshooting first software comprising:

in response to an error in operation of the first software on a first platform, establishing a connection between the first platform and a remote platform; and

where the error corresponds to known error data from the remote platform, overwriting a portion of the first software with updated code.

16. The method of claim 15 further comprising downloading the known error data from the remote platform to the first platform.

17. The method of claim 15 wherein establishing the connection comprises establishing an Internet connection.

18. The method of claim 17 further comprising determining whether access to the Internet is available.

19. The method of claim 15 further comprising obtaining authorization from a user of the first software to establish the connection.

20. The method of claim 19 further comprising,

where authorization is not obtained, providing software support information to the user of the first software.

21. The method of claim 15 further comprising shutting down operation of the first software before overwriting the first portion of the first software.

22. The method of claim 15 further comprising, after overwriting the first portion of the first software, running a software diagnostic routine on the first software.

23. The method of claim 15 further comprising, where the error does not correspond to the known error data, uploading data files associated with the first software from the first platform to the remote platform via the connection; and

reproducing the error on the remote platform using the data files.

24. The method of claim 23 further comprising obtaining authorization from a user of the first software for uploading the data files.

25. The method of claim 24 further comprising, where authorization is not obtained, providing software support information to the user of the first software.

26. The method of claim 23 further comprising, before uploading the data files, compiling information from the first platform regarding the data files.

27. The method of claim 15 further comprising, where the error does not correspond to the known error data, operating the first software on the first platform from the remote platform via the connection, thereby reproducing the error on the first platform.

28. The method of claim 27 further comprising

obtaining authorization from a user of the first software for operating the first software from the remote platform.

29. The method of claim 28 further comprising, where authorization is not obtained, providing software support information to the user of the first software.

30. At least one computer readable medium containing program instructions for troubleshooting first software, said at least one computer readable medium comprising:

computer readable code for, in response to an error in operation of the first software on a first platform, establishing a connection between the first platform and a remote platform; and

computer readable code for, where the error corresponds to known error data from the remote platform, overwriting a portion of the first software with updated code.

31. A computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by at least one processor, cause said at least one processor to troubleshoot first software by:

in response to an error in operation of the first software on a first platform, establishing a connection between the first platform and a remote platform; and

where the error corresponds to known error data from the remote platform, overwriting a portion of the first software with updated code.

32. A computer readable medium containing program instructions corresponding to first software, the first software having been troubleshot according to a method comprising:

in response to an error in operation of the first software on a first platform, establishing a connection between the first platform and a remote platform; and

where the error corresponds to known error data from

the remote platform, overwriting a portion of the first software with updated code.

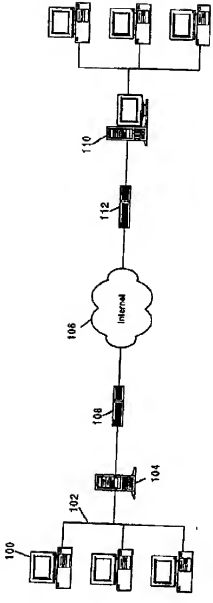


Fig. 1

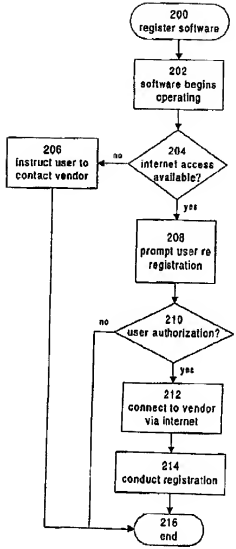
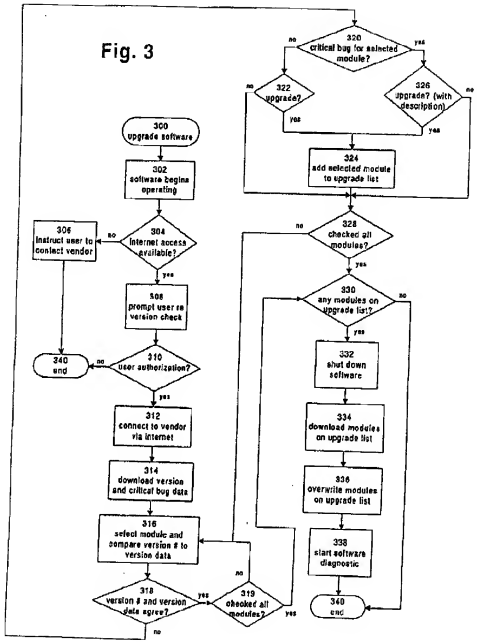

200
register software
202
software begins operating
204
internet access available?
no
206
instruct user to contact vendor
yes
208
prompt user re registration
210
user authorization?
no
yes
212
connect to vendor via internet
214
conduct registration
216
end

Fig. 2

Fig. 3


300
upgrade software
302
software begins operating
304
internet access available?
306
instruct user to contact vendor
308
prompt user re version check
310
user authorization?
340
end
312
connect to vendor via internet
314
download version and critical bug data
316
select module and compare version # to version data
318
version # and version data agree?
319
checked all modules?
320
critical bug for selected module?
322
upgrade?
326
upgrade? (with description)
324
add selected module to upgrade list
328
checked all modules?
330
any modules on upgrade list?
332
shut down software
334
download modules on upgrade list
336
overwrite modules on upgrade list
338
start software diagnostic
340
end
yes
no

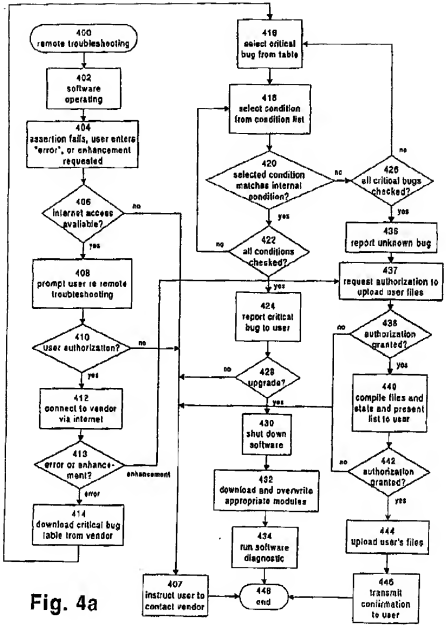

400
remote troubleshooting
402
software operating
404
assertion fails, user enters "error", or enhancement requested
406
internet access available?
no
yes
408
prompt user re remote troubleshooting
410
user authorization?
no
yes
412
connect to vendor via internet
413
error or enhancement?
enhancement
error
414
download critical bug table from vendor
407
instruct user to contact vendor
416
select critical bug from table
418
select condition from condition list
420
selected condition matches internal condition?
no
yes
422
all conditions checked?
no
yes
424
report critical bug to user
428
upgrade?
no
yes
430
shut down software
432
download and overwrite appropriate modules
434
run software diagnostic
448
end
426
all critical bugs checked?
no
yes
436
report unknown bug
437
request authorization to upload user files
438
authorization granted?
no
yes
440
compile files and stats and present list to user
442
authorization granted?
no
yes
444
upload user's files
445
transmit confirmation to user

Fig. 4a

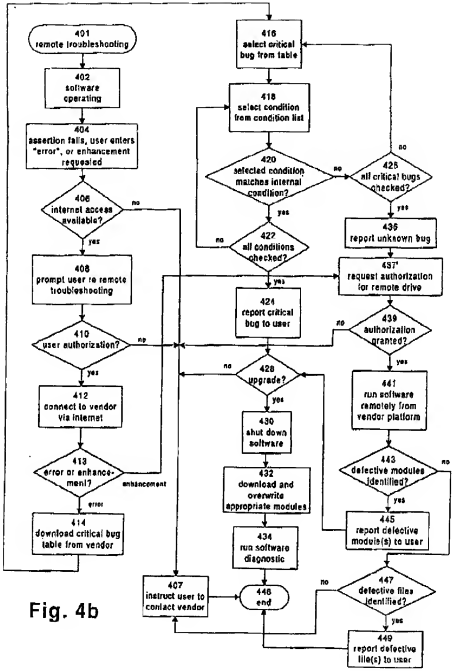

401 remote troubleshooting
402 software operating
404 assertion fails, user enters "error", or enhancement requested
406 internet access available?
no
yes
408 prompt user re remote troubleshooting
410 user authorization?
no
yes
412 connect to vendor via internet
413 error or enhancement?
enhancement
error
414 download critical bug table from vendor
407 instruct user to contact vendor
416 select critical bug from table
418 select condition from condition list
420 selected condition matches internal condition?
no
yes
426 all critical bugs checked?
no
yes
435 report unknown bug
427 all conditions checked?
no
yes
437 request authorization for remote drive
424 report critical bug to user
439 authorization granted?
no
yes
428 upgrade?
no
yes
441 run software remotely from vendor platform
430 shut down software
443 defective modules identified?
no
yes
432 download and overwrite appropriate modules
445 report defective module(s) to user
434 run software diagnostic
447 defective files identified?
no
yes
446 end
449 report defective file(s) to user

Fig. 4b

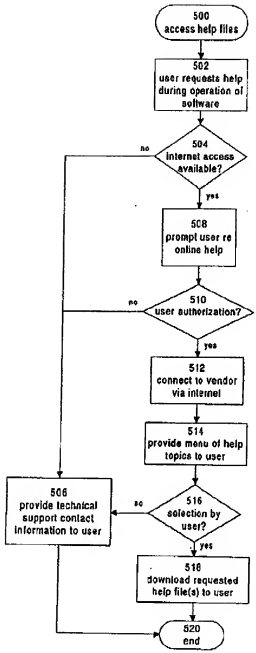

500
access help files
502
user requests help during operation of software
504
internet access available?
no
yes
508
prompt user re online help
510
user authorization?
no
yes
512
connect to vendor via internet
514
provide menu of help topics to user
516
selection by user?
no
yes
506
provide technical support contact information to user
518
download requested help file(s) to user
520
end

Fig. 5

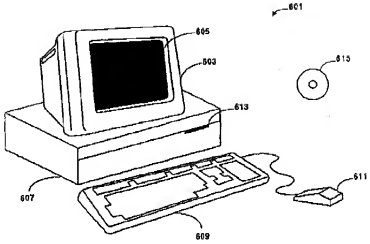

601
605
603
615
613
611
607
609

FIG. 6

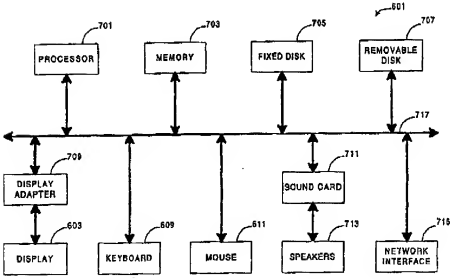

601
701
PROCESSOR
703
MEMORY
705
FIXED DISK
707
REMOVABLE DISK
717
709
DISPLAY ADAPTER
711
SOUND CARD
603
DISPLAY
609
KEYBOARD
611
MOUSE
713
SPEAKERS
715
NETWORK INTERFACE

FIG. 7

METHOD FOR PROVIDING REMOTE SOFTWARE TECHNICAL SUPPORT

ABSTRACT OF THE DISCLOSURE

A method for upgrading software is described. According to the invention, in response to operation of the software on a first platform, a connection is established between the first platform and a remote platform. Where first version data associated with the software are different from second version data from the remote platform, a portion of the software is overwritten with first updated code from the remote platform.